APR 1 2 2004
PATENT & TRADEMARK OFFICE

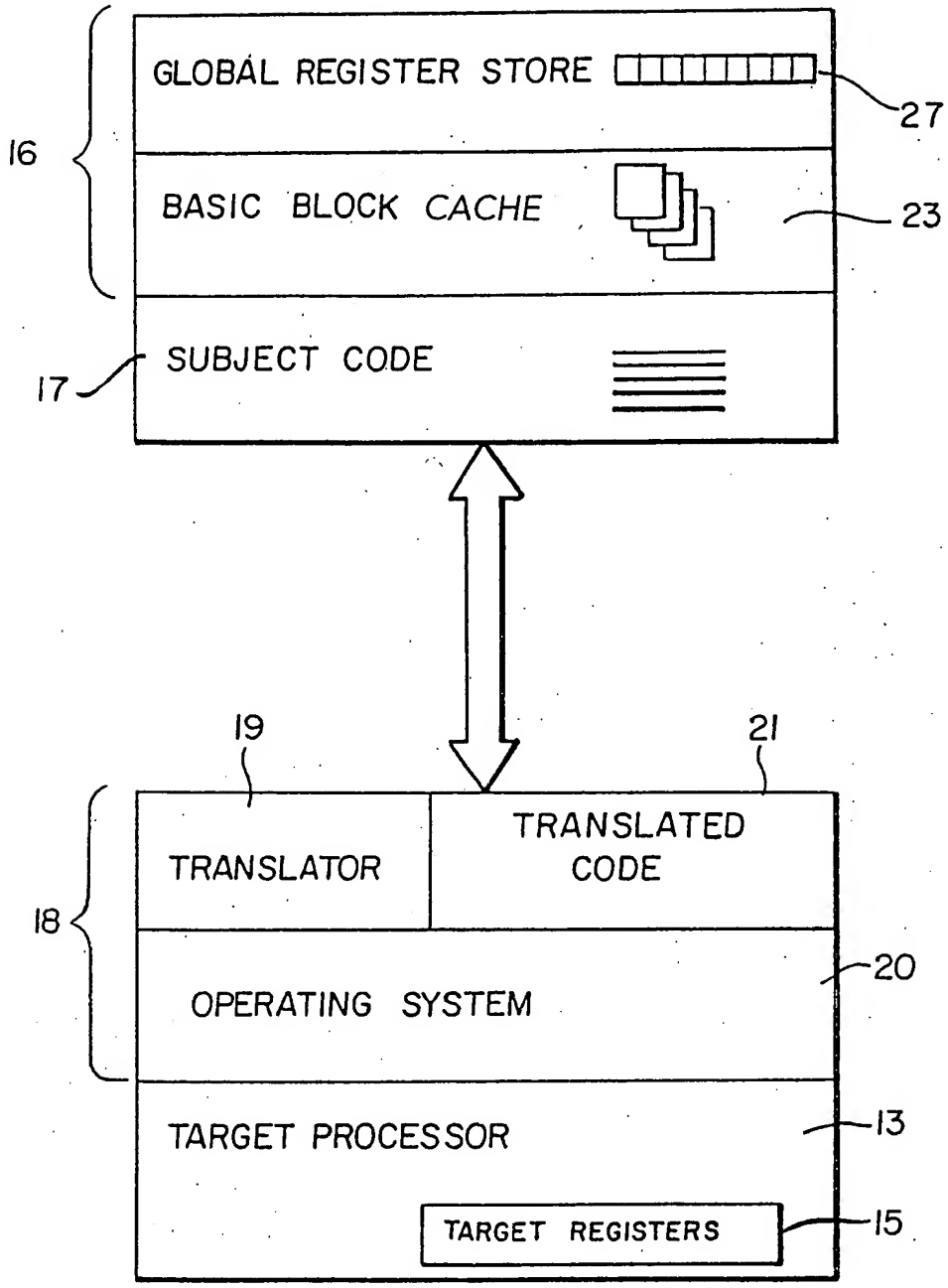


FIG. 1

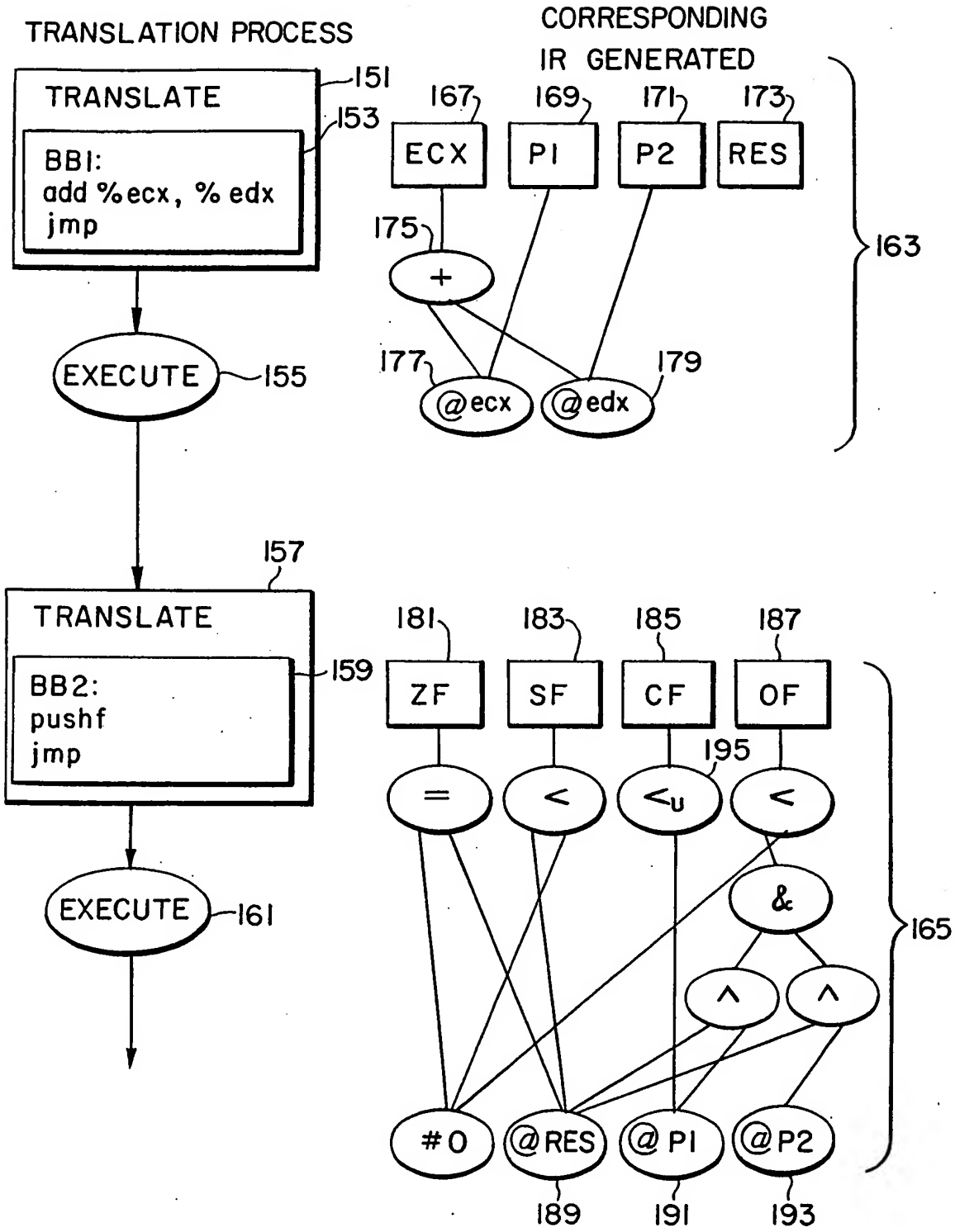

TRANSLATION PROCESS
CORRESPONDING IR GENERATED
TRANSLATE
151
153
BB1:
add %ecx, % edx
jmp
EXECUTE
155
167
169
171
173
ECX
P1
P2
RES
175
+
177
179
@ecx
@edx
163
157
TRANSLATE
159
BB2:
pushf
jmp
EXECUTE
161
181
183
185
187
ZF
SF
CF
OF
195
=
<
<u
<
&
^
^
#0
@RES
@P1
@P2
165
189
191
193

FIG. 2

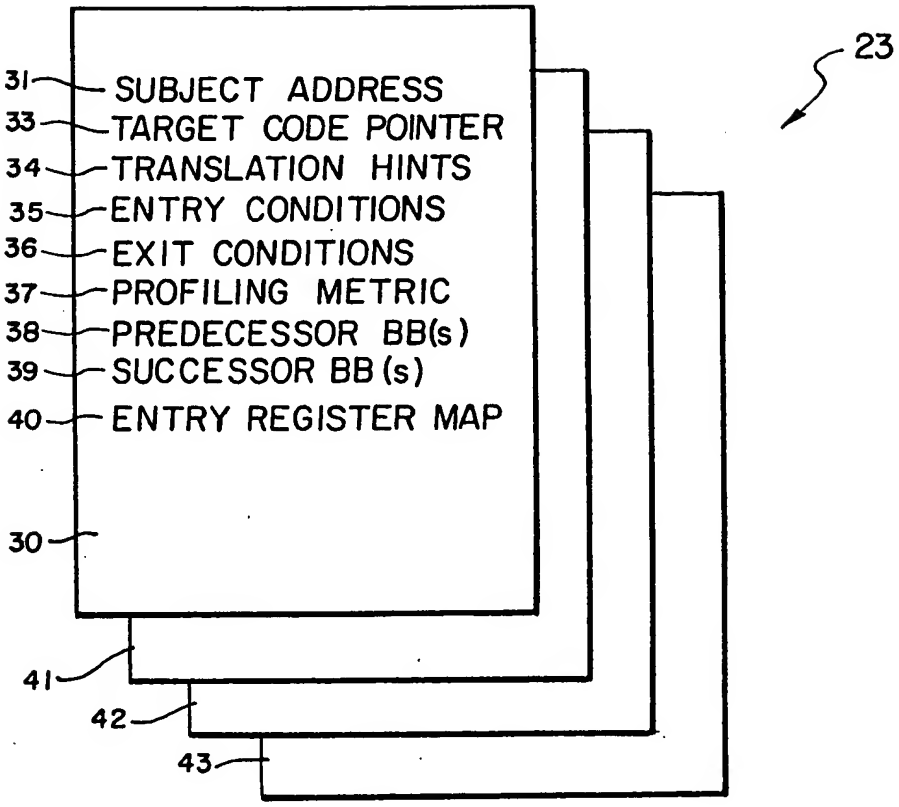

23
31
SUBJECT ADDRESS
33
TARGET CODE POINTER
34
TRANSLATION HINTS
35
ENTRY CONDITIONS
36
EXIT CONDITIONS
37
PROFILING METRIC
38
PREDECESSOR BB(s)
39
SUCCESSOR BB(s)
40
ENTRY REGISTER MAP
30
41
42
43

FIG. 3

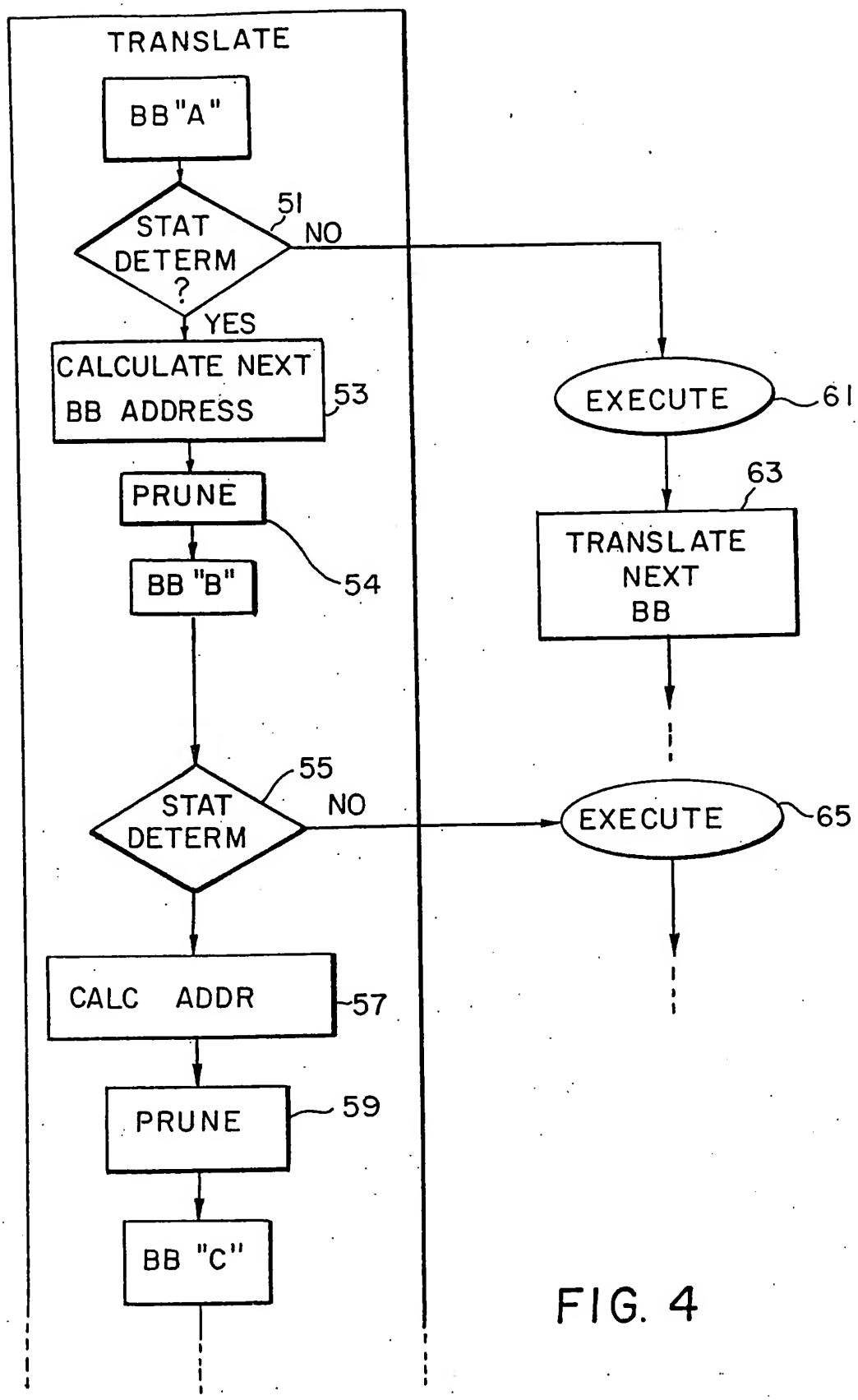

TRANSLATE
BB "A"
STAT DETERM ?
51
NO
YES
CALCULATE NEXT BB ADDRESS
53
PRUNE
54
BB "B"
55
STAT DETERM
NO
CALC ADDR
57
PRUNE
59
BB "C"
EXECUTE
61
63
TRANSLATE NEXT BB
EXECUTE
65

FIG. 4

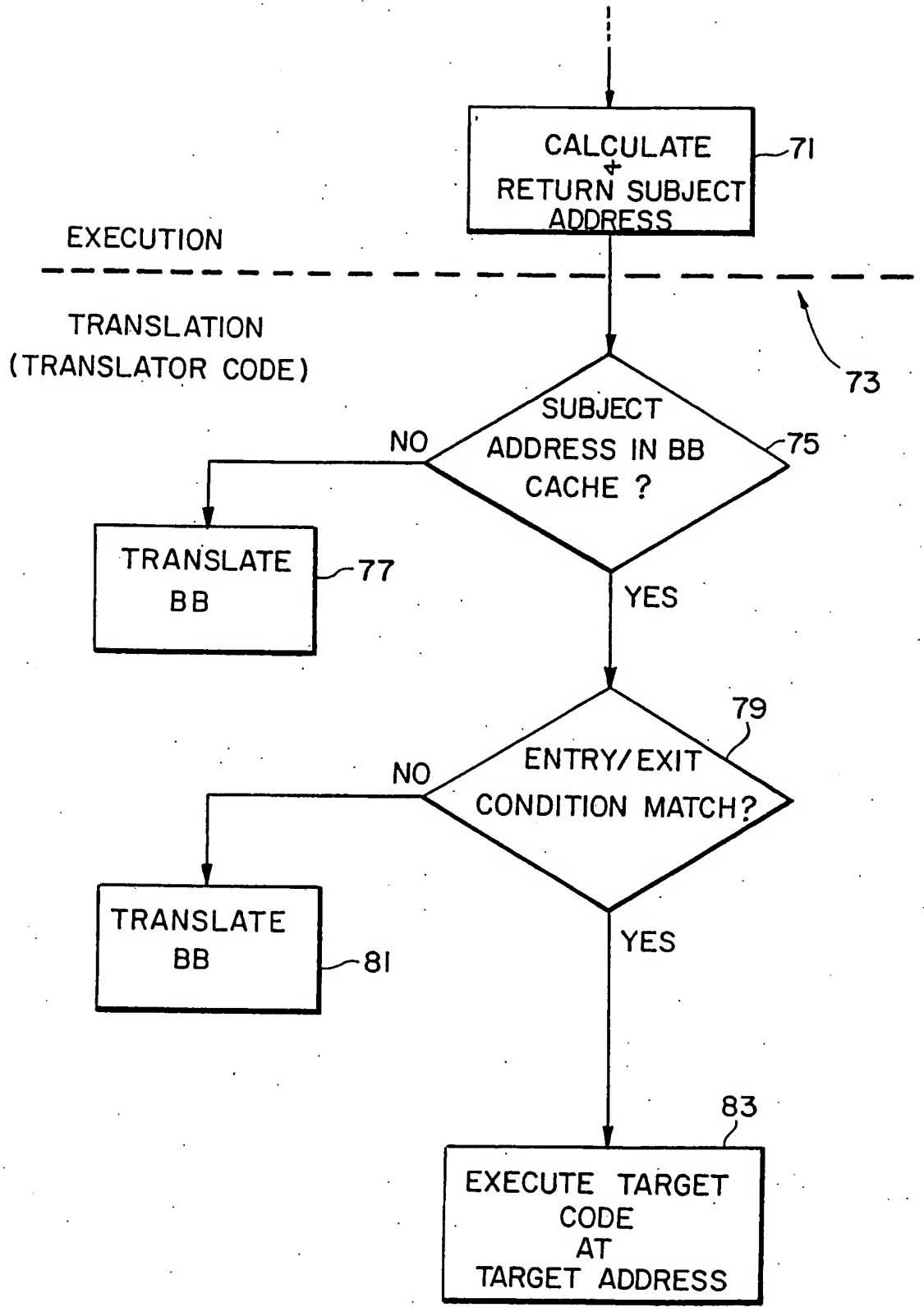

CALCULATE
RETURN SUBJECT ADDRESS
71
EXECUTION
TRANSLATION
(TRANSLATOR CODE)
73
SUBJECT ADDRESS IN BB CACHE ?
75
NO
YES
TRANSLATE BB
77
ENTRY/EXIT CONDITION MATCH?
79
NO
YES
TRANSLATE BB
81
EXECUTE TARGET CODE AT TARGET ADDRESS
83

FIG. 5

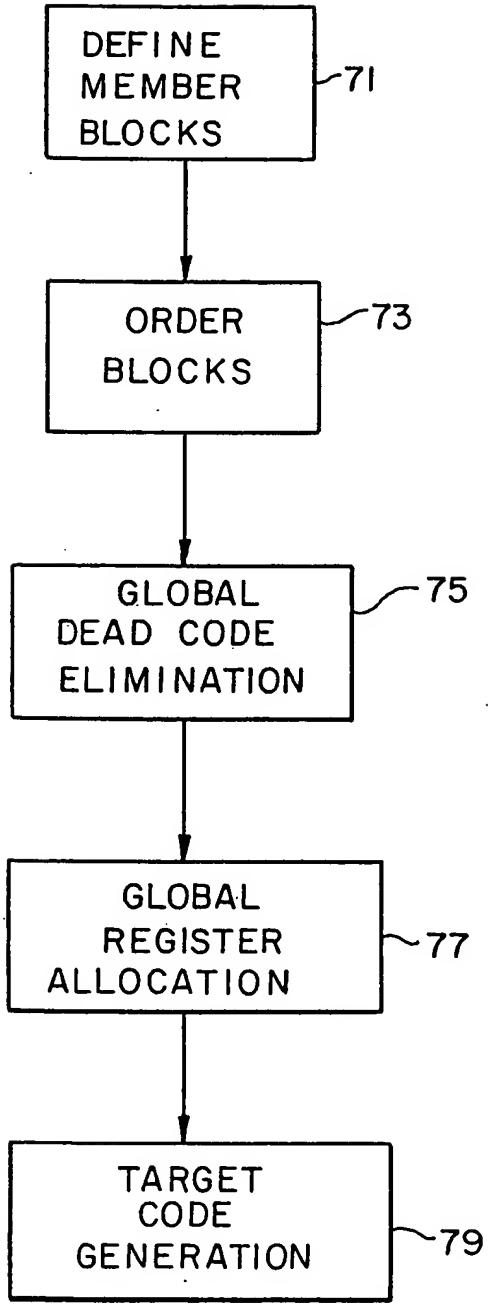

DEFINE MEMBER BLOCKS
71
ORDER BLOCKS
73
GLOBAL DEAD CODE ELIMINATION
75
GLOBAL REGISTER ALLOCATION
77
TARGET CODE GENERATION
79

FIG. 6

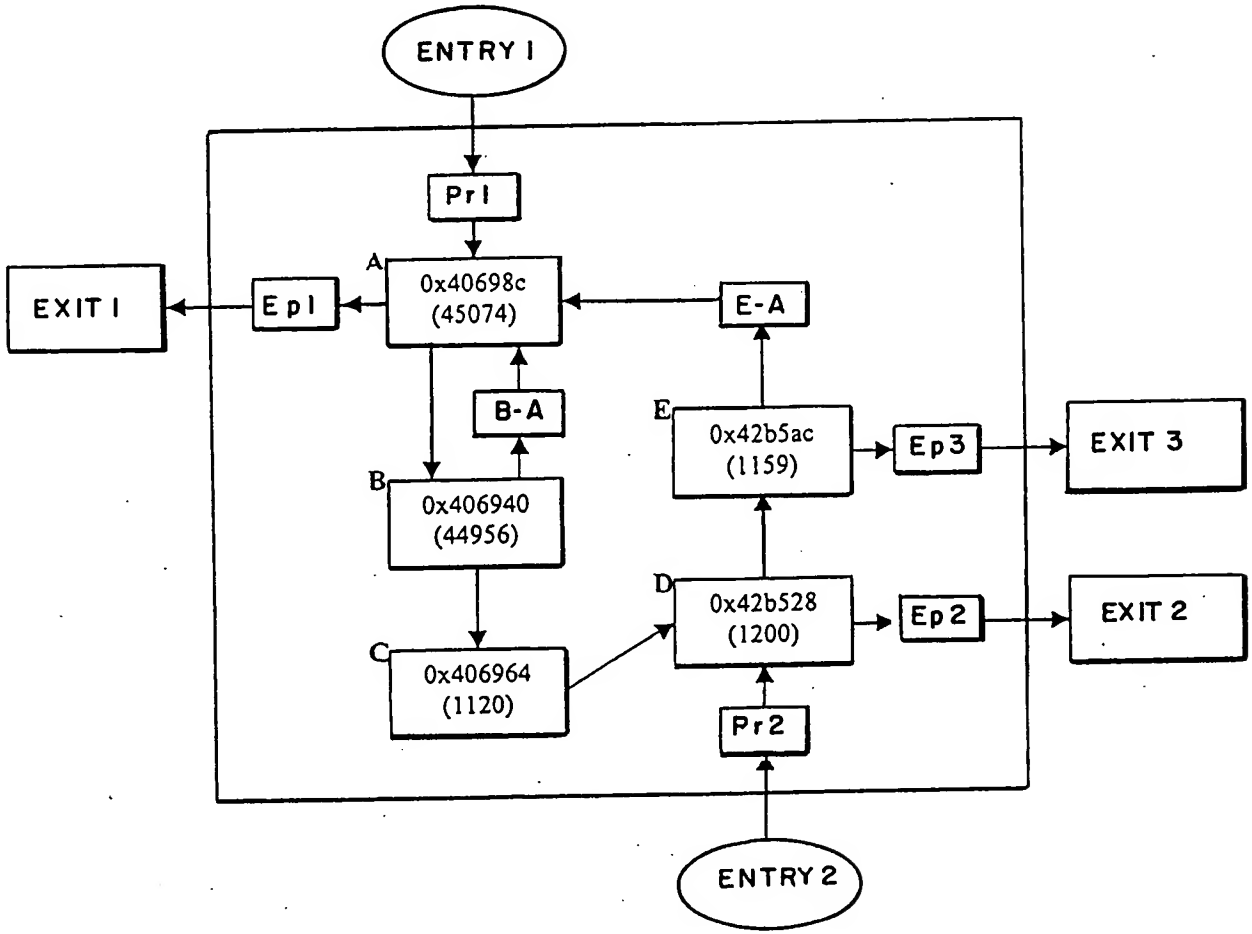

ENTRY 1
Pr1
A
0x40698c
(45074)
EXIT 1
Ep1
E-A
B-A
B
0x406940
(44956)
E
0x42b5ac
(1159)
Ep3
EXIT 3
D
0x42b528
(1200)
Ep2
EXIT 2
C
0x406964
(1120)
Pr2
ENTRY 2

FIG. 7

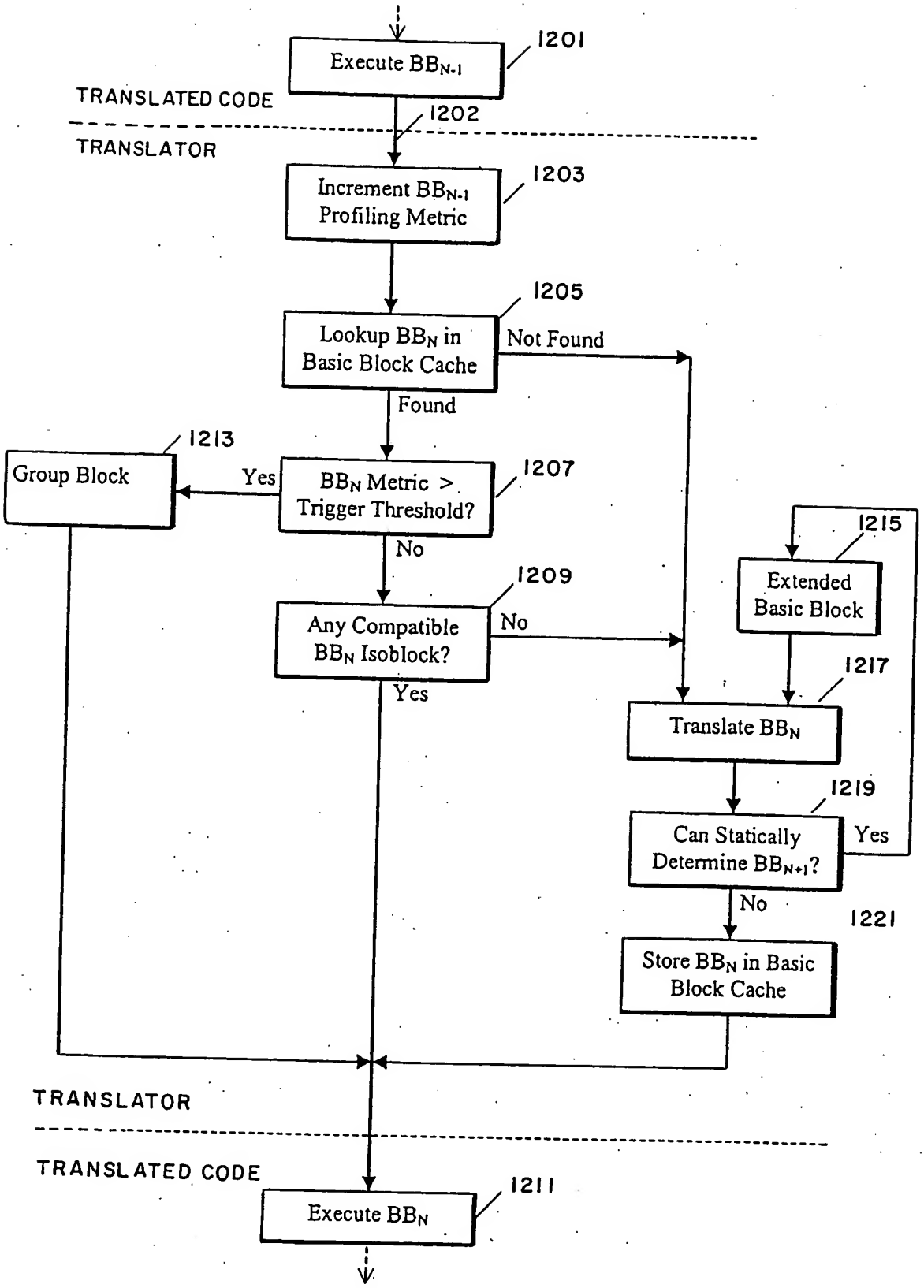

Execute BB_{N-1}
1201
TRANSLATED CODE
1202
TRANSLATOR
Increment BB_{N-1} Profiling Metric
1203
Lookup BB_N in Basic Block Cache
1205
Not Found
Found
Group Block
1213
Yes
BB_N Metric > Trigger Threshold?
1207
No
Any Compatible BB_N Isoblock?
1209
No
Yes
1215
Extended Basic Block
1217
Translate BB_N
1219
Can Statically Determine BB_{N+1}?
Yes
No
1221
Store BB_N in Basic Block Cache
TRANSLATOR
TRANSLATED CODE
Execute BB_N
1211

FIG. 8

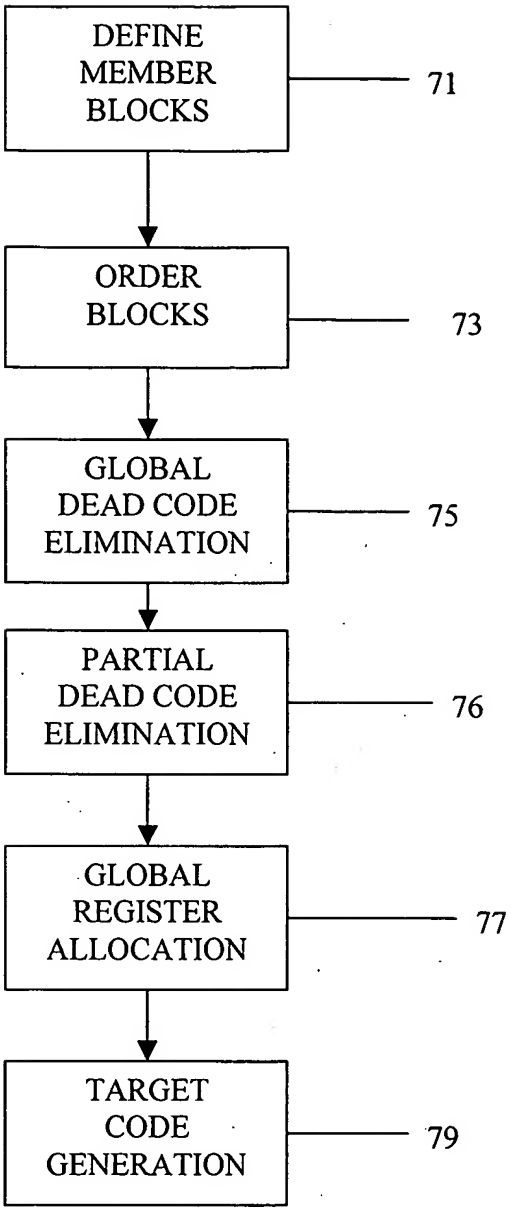

DEFINE MEMBER BLOCKS
71
ORDER BLOCKS
73
GLOBAL DEAD CODE ELIMINATION
75
PARTIAL DEAD CODE ELIMINATION
76
GLOBAL REGISTER ALLOCATION
77
TARGET CODE GENERATION
79

FIG. 9

FIG. 10A

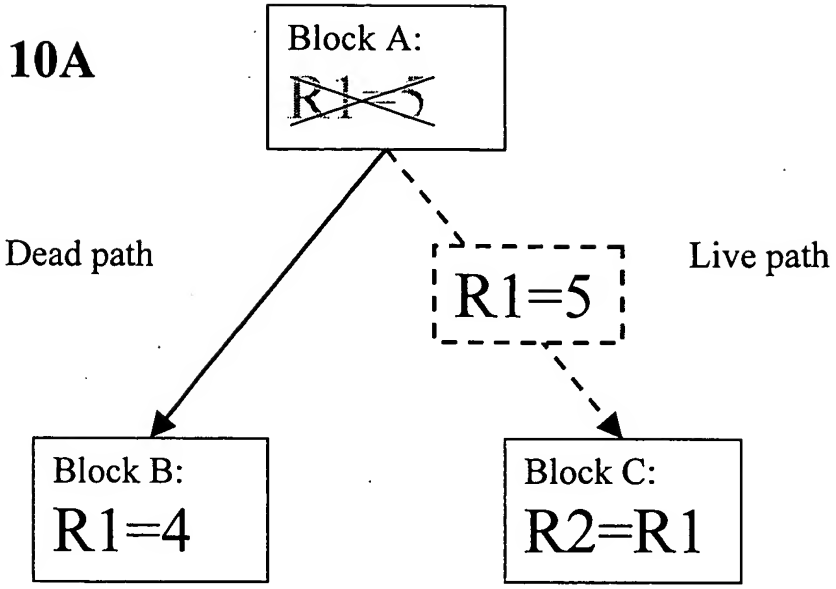
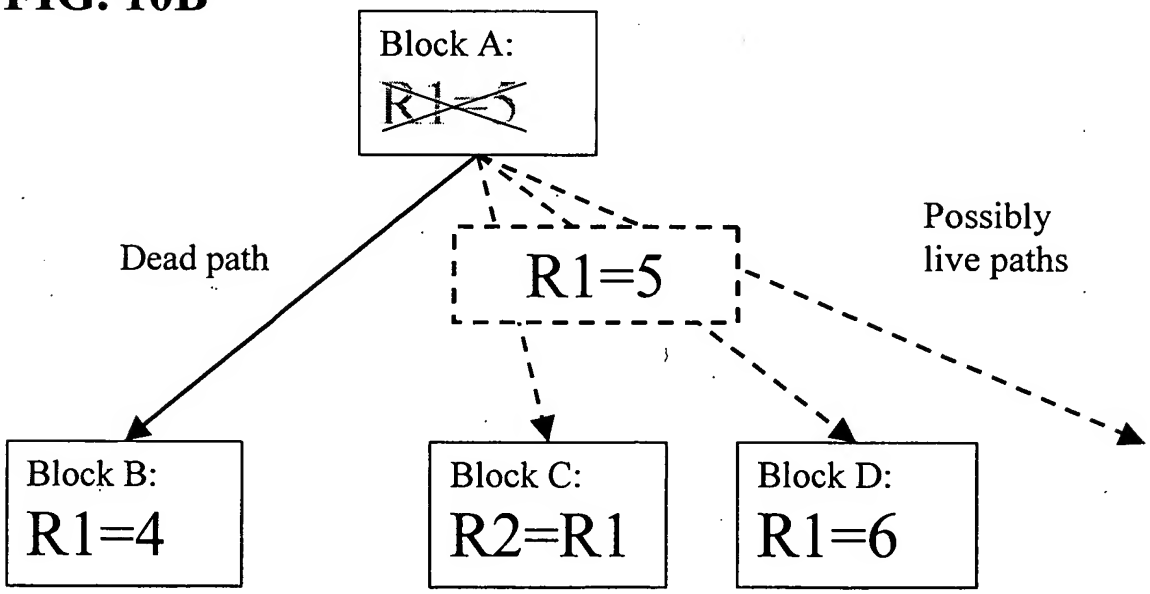


FIG. 10B



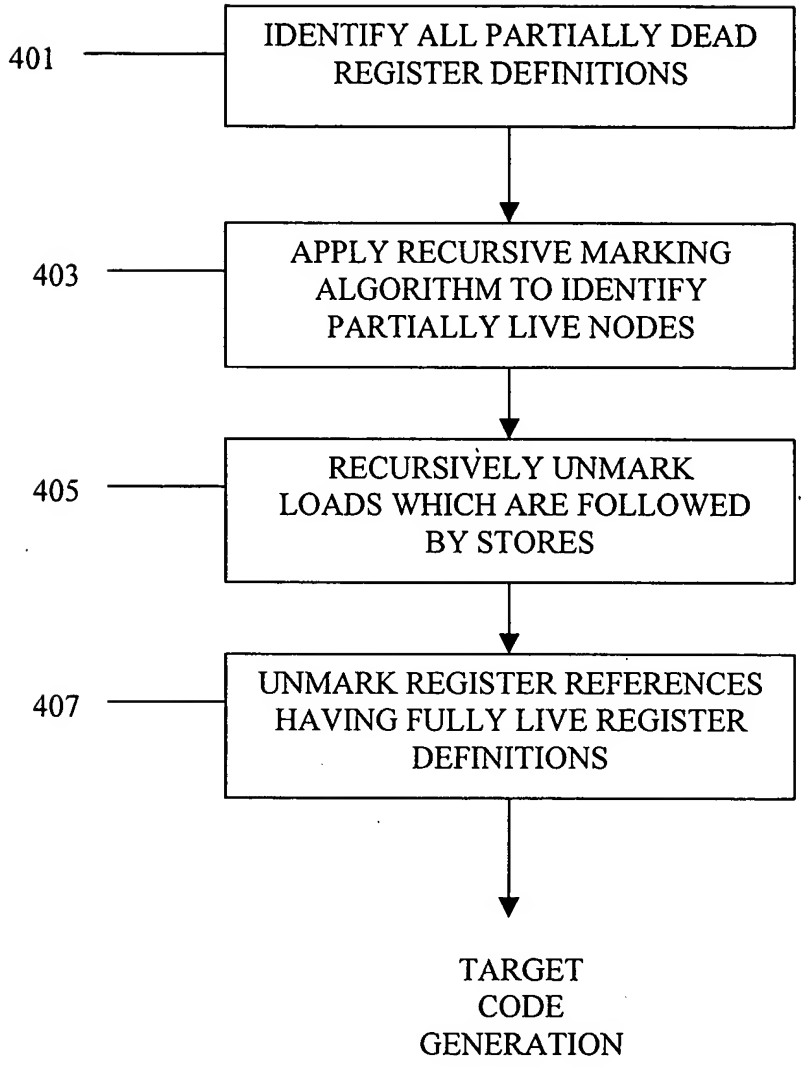

401
IDENTIFY ALL PARTIALLY DEAD REGISTER DEFINITIONS
403
APPLY RECURSIVE MARKING ALGORITHM TO IDENTIFY PARTIALLY LIVE NODES
405
RECURSIVELY UNMARK LOADS WHICH ARE FOLLOWED BY STORES
407
UNMARK REGISTER REFERENCES HAVING FULLY LIVE REGISTER DEFINITIONS
TARGET CODE GENERATION

FIG. 11

FIG. 12

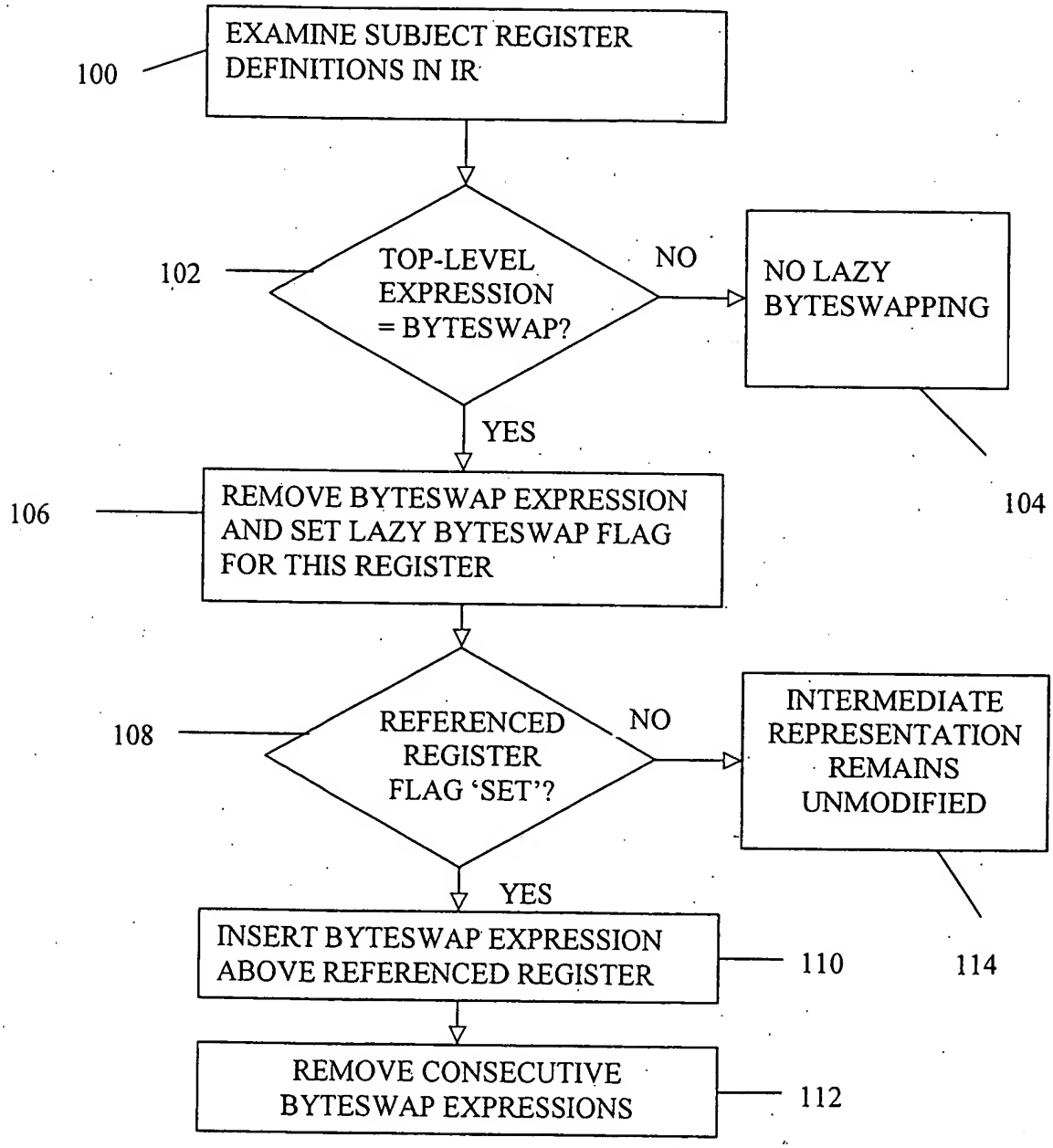

EXAMINE SUBJECT REGISTER DEFINITIONS IN IR
100
TOP-LEVEL EXPRESSION = BYTESWAP?
102
NO
NO LAZY BYTESWAPPING
104
YES
REMOVE BYTESWAP EXPRESSION AND SET LAZY BYTESWAP FLAG FOR THIS REGISTER
106
REFERENCED REGISTER FLAG 'SET'?
108
NO
INTERMEDIATE REPRESENTATION REMAINS UNMODIFIED
114
YES
INSERT BYTESWAP EXPRESSION ABOVE REFERENCED REGISTER
110
REMOVE CONSECUTIVE BYTESWAP EXPRESSIONS
112

FIG. 13A

Subject Code 200

```
LABEL1    r3 := LOAD(r2 + 24)
          r1 := r1 -1
          CMP r1 0
          BC LABEL1
          STORE 16(r5) r3
```

Group Block

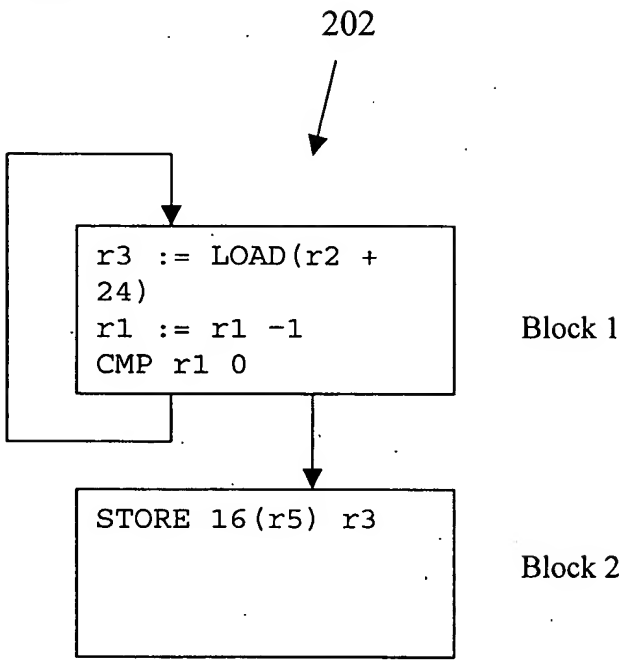
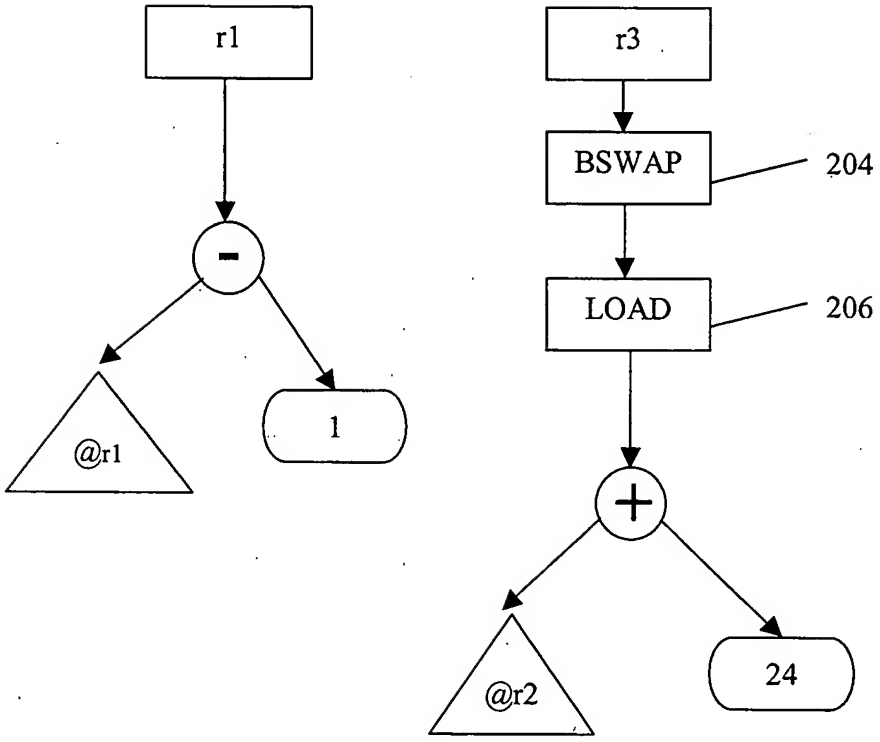


FIG. 13B

Block 1 IR Without Lazy Byteswapping



Block 2 IR Without Lazy Byteswapping

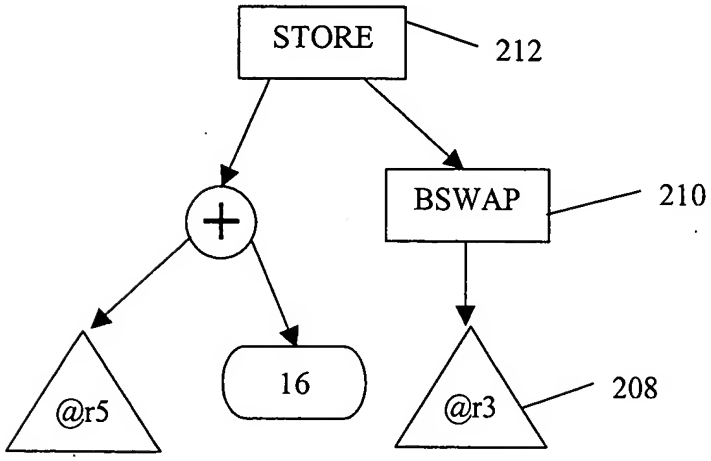
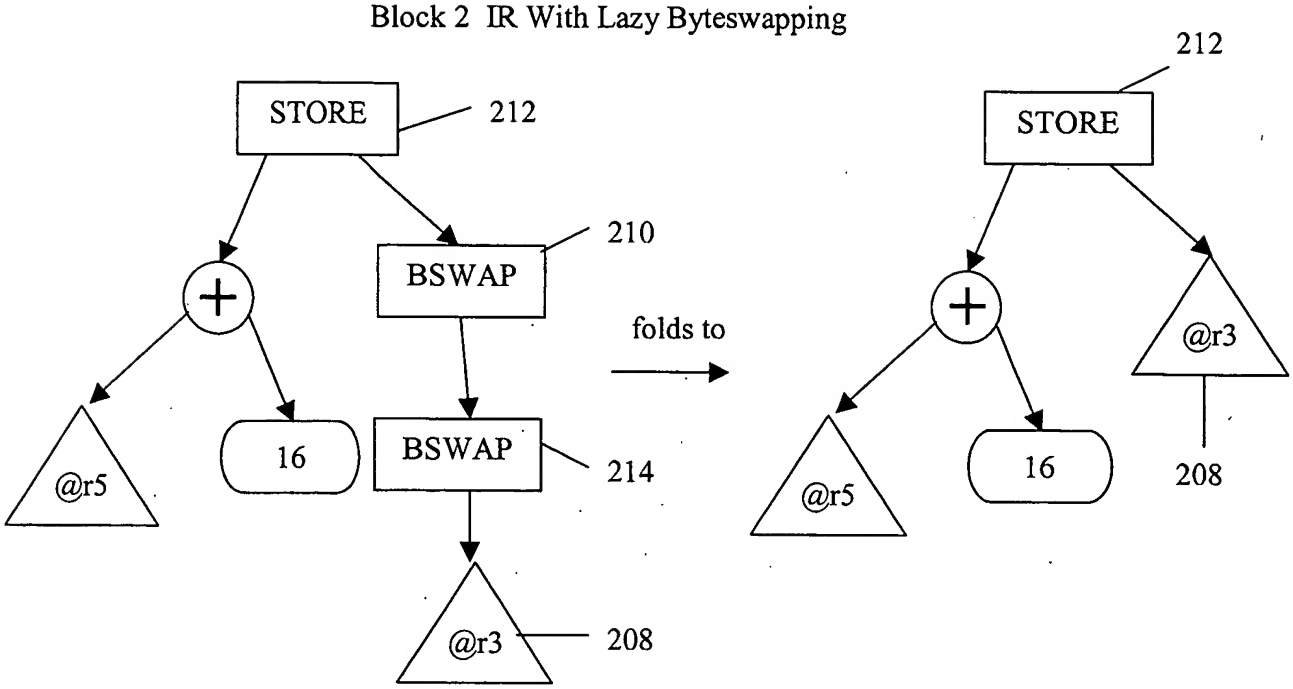


FIG. 13C


Block 2 IR With Lazy Byteswapping
STORE
212
BSWAP
210
+
@r5
16
BSWAP
214
@r3
208
folds to
212
STORE
+
@r5
16
@r3
208

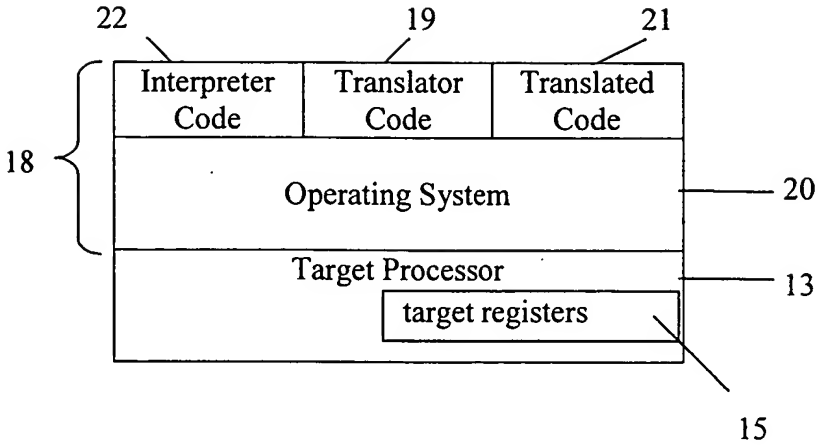

22
19
21
Interpreter Code
Translator Code
Translated Code
18
Operating System
20
Target Processor
13
target registers
15

FIG. 14

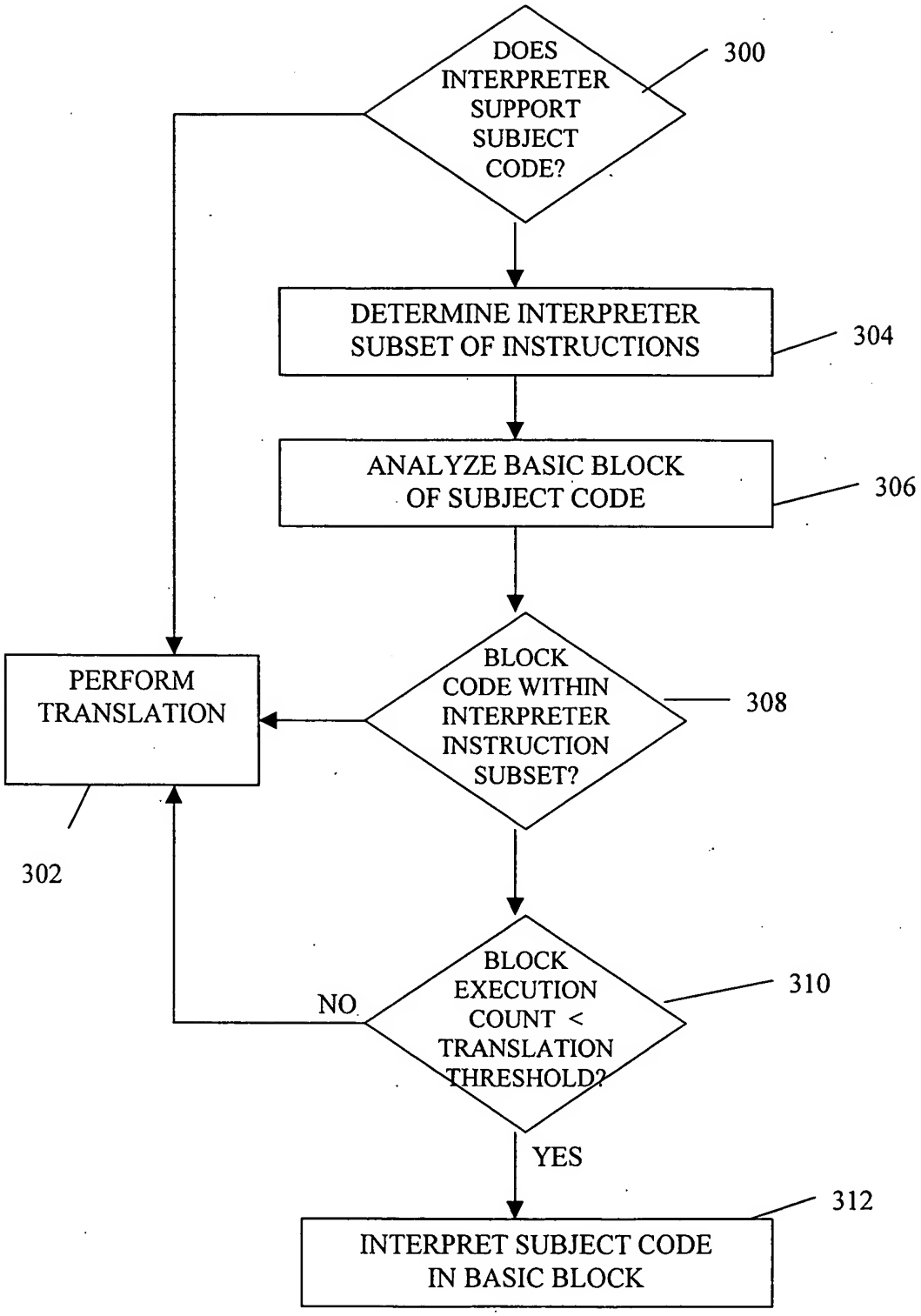

DOES INTERPRETER SUPPORT SUBJECT CODE?
300
DETERMINE INTERPRETER SUBSET OF INSTRUCTIONS
304
ANALYZE BASIC BLOCK OF SUBJECT CODE
306
BLOCK CODE WITHIN INTERPRETER INSTRUCTION SUBSET?
308
PERFORM TRANSLATION
302
BLOCK EXECUTION COUNT < TRANSLATION THRESHOLD?
310
NO
YES
INTERPRET SUBJECT CODE IN BASIC BLOCK
312

FIG. 15